International Journal of Electrical, Electronics and Computer Systems
ISSN: 2347-2820
Volume 15 Issue 1s (2026) | Pages: 212-216

An AI-Powered Tool for Automated Web Application Penetration Testing using PPO

Anirudh Patki[1], Prathamesh Jadhav[2], Gopal Tayade[3], Alka Kumbhar[4]

[1,2,3,4] Department of Computer Science Engineering, Genba Sopanrao Moze College of Engineering, Pune

Peer Review Information

***Type**: Article*
***Received**: 13 February 2026*
***Revised**: 14 March 2026*
***Accepted**: 15 April 2026*
***Published**: 21 May 2026*

Abstract

This paper proposes an autonomous platform for web penetration testing designed to overcome the limitations of manual testing. The core of this system is a Reinforcement Learning (RL) Automation Core (L2), which acts as the high-level "Strategist." This PPO-based agent learns the optimal sequential workflow of a penetration test. A primary innovation is the Localized Intelligence Layer (L1), which utilizes a Large Language Model (LLM) run locally via Ollama. This privacy-first "Expert" is called by the agent's tools to perform strategic analysis, such as identifying vulnerability sinks from code. This architecture is supported by a Hybrid AI Detection Engine (L3) that synergizes Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) performed by custom scripts using a headless browser to validate exploits. The platform focuses on identifying and validating the OWASP Top 10 vulnerabilities. All findings, Proof-of-Concept (PoC) steps, and remediation guidance are generated in a final command-line report. The system is designed to target high detection accuracy (>=90%) and significantly reduce the false positives that plague traditional scanners.

Keywords: Artificial Intelligence; Cybersecurity; Deep Reinforcement Learning; Large Language Models; OLLAMA; Penetration Testing.

How to Cite This Article

Patki, A., Jadhav, P., Tayade, G., Kumbhar, A. (2026). An AI-Powered Tool for Automated Web Application Penetration Testing using PPO. *International Journal of Electrical, Electronics and Computer Systems*, 15(1s), 212-216.

Introduction

The rapidly accelerating pace of digital transformation has led to a critical imbalance in the cybersecurity landscape: the gap between the speed at which new software vulnerabilities appear and the ability of human security professionals to detect and remediate them is widening. This urgent necessity for continuous, autonomous, and intelligent penetration testing (pentesting) solutions drives the research presented in this paper, which details a novel hybrid AI framework for web vulnerability discovery and exploitation. Early attempts to automate offensive security primarily relied on deterministic scripting frameworks built around tools like Metasploit. While effective against known attack patterns and predefined configurations, these first-generation automation systems are fundamentally rigid. Their reliance on hard-coded expert rules renders them incapable of adapting to novel environments, incomplete reconnaissance data, or previously unseen system configurations, limiting their real-world efficacy.

A subsequent line of research introduced Reinforcement Learning (RL), which models pentesting as a sequential decision-making process in partially observable environments. However, pure RL agents often require a heavily simplified action space for tractability and necessitate a massive number of costly training interactions that rarely transfer effectively between distinct network environments, resulting in modest coverage and integration complexity. These limitations underscore that purely Deep Learning (DL) or RL-based automation is insufficient for addressing the dynamic and multi-phase nature inherent to a complete penetration test. To overcome the challenges of previous automation paradigms, recent research has pivoted toward advanced AI agent-based models. These systems, such as RapidPen and those utilizing Hierarchical Planning (HPTSA), employ specialized Large Language Models (LLMs) that collaborate to emulate the comprehensive strategic workflow of human red teams. The LLM's core strengths — language understanding, complex strategic planning, and sophisticated tool interaction — are leveraged to achieve goals like initial access with high reliability and speed.

Literature Review

The application of artificial intelligence to automate penetration testing has evolved through several distinct phases, each increasing in abstraction and capability. This progression began with machine learning models for augmenting simple tasks, advanced to reinforcement learning for strategic pathfinding, and has most recently incorporated large language models for human-like reasoning.

Initial forays into AI-driven offensive security focused on applying classical machine learning (ML) to enhance the performance of traditional scanning tools. These early systems aimed to improve pattern recognition beyond simple signature matching. A notable example is the GyoiThon tool, which utilized the Naive Bayes algorithm to analyze server configurations and match them against known vulnerability databases. This approach proved more effective than the default scanning modes of many tools, demonstrating that even simple ML models could provide a tangible improvement in vulnerability discovery by moving beyond basic pattern matching.

A significant paradigm shift occurred with the application of Reinforcement Learning (RL), which reframed penetration testing from a series of independent scans into a complex, sequential decision-making process. This approach enabled the automation of strategic pathfinding, where an agent learns the optimal sequence of actions to achieve a goal. The most recent evolutionary step has been the integration of Large Language Models (LLMs), which introduce human-like reasoning, vast knowledge synthesis, and sophisticated tool orchestration into the automation loop. LLMs address the highest level of abstraction: strategic intent. The PENTESTGPT framework exemplifies this approach, employing a multi-agent architecture where different LLM-powered modules collaborate on reasoning, command generation, and output parsing.

A critical review of the literature reveals a clear "chain of abstraction" in the evolution of AI for penetration testing: ML addressed pattern recognition, RL addressed sequential action, and LLMs are now addressing strategic intent. While each paradigm has shown success in its respective domain, existing systems tend to specialize in only one of these areas. This specialization creates a significant research gap: the need for a unified framework that integrates all three levels of abstraction to create a truly autonomous agent that can reason, plan, and act with high fidelity.

***Table I**: Comparative Analysis of Automated Pentesting Frameworks*

Framework	**Core AI Engine**	**Planning / Strategy Model**	**Detection Model**	**Privacy Architecture**
Proposed Framework	Hybrid (LLM + RL)	LLM Strategic Planner (Local)	Hybrid (SAST + DAST + ML)	Privacy-First (OLLAMA)

PENTESTGPT	LLM (Multi-Agent)	LLM-Based Reasoning (Cloud API)	Relies on External Tool Output	Cloud-Dependent
DeepExploit / Shennina	Deep Reinforcement Learning (A3C)	RL Policy Network	Relies on External Tool Output	Not Specified (Typically Local Agent)
PenBox	Reinforcement Learning (DQN)	RL Policy Network	Relies on External Tool Output	Not Specified (Typically Local Agent)

Methodology

The proposed framework is designed as a multi-layered, hierarchical system that emulates and enhances the cognitive workflow of an expert security analyst. It synergistically combines a high-level strategic planner with a tactical execution core and a rigorous validation engine to create a fully autonomous, command-line-driven penetration testing solution.

System Architecture

The architecture of the proposed model is built upon three specialized, interconnected AI components, each addressing a distinct level of the penetration testing process. This hierarchical structure allows for a clear separation of concerns, where high-level strategy guides tactical execution, and all findings are rigorously validated before being actioned. L2 Reinforcement Learning (RL) Automation Core (Strategic Planner): This layer serves as the cognitive core of the framework, responsible for high-level strategic planning and goal- driven decision-making. It employs a Proximal Policy Optimization (PPO) agent that observes the environment's state (e.g., recon_done, vulnerabilities_found) and selects the next high-level action (e.g., WEB_CRAWL), acting as the primary "Strategist". L1 Localized Intelligence Layer (Tactical Expert): This layer is the "on-demand" expert, responsible for tactical execution and deep analysis. Its cornerstone is a Large Language Model (LLM) fine-tuned on cybersecurity data and deployed locally using the OLLAMA framework.

It is called by the tools to perform specific tasks such as static code analysis, vulnerability identification, or exploit generation. L3 Hybrid AI Detection Engine (Validator): This final layer acts as the system's quality control and validation mechanism. To address the pervasive industry problem of "alert fatigue," this engine fuses outputs from Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) to cross-validate all findings.

Implementation

The practical implementation is centered around three key areas:

Model Fine-Tuning and Data Corpus: The base LLM is fine-tuned using the OLLAMA framework on a comprehensive cybersecurity data corpus including vulnerability databases (CVE, CWE, NVD), intentionally vulnerable applications (DVWA, OWASP Juice Shop), exploit code from ExploitDB, and open-source repositories from GitHub and GitLab. Execution and Orchestration Layer: The technical backbone is an orchestration layer developed in Python. This layer consists of custom Python wrappers that create a standardized programmatic interface for penetration testing tools including Nmap, sqlmap, Gobuster, and Metasploit. Software and Environment Requirements: The system requires Python 3 for core orchestration logic and AI model interaction. Docker is utilized to containerize the various penetration testing tools, ensuring isolation, portability, and a consistent operating environment.

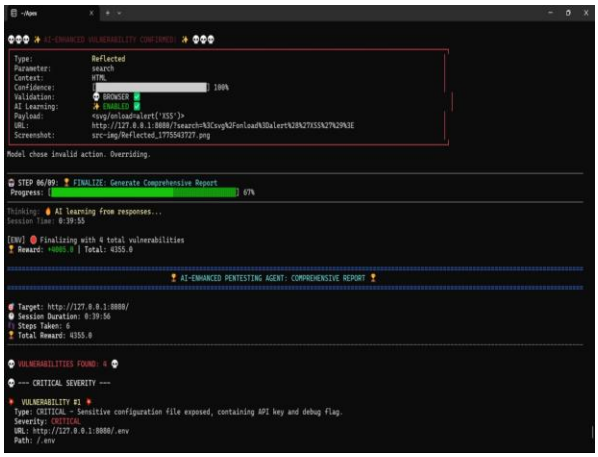

***Fig** 1. Output*

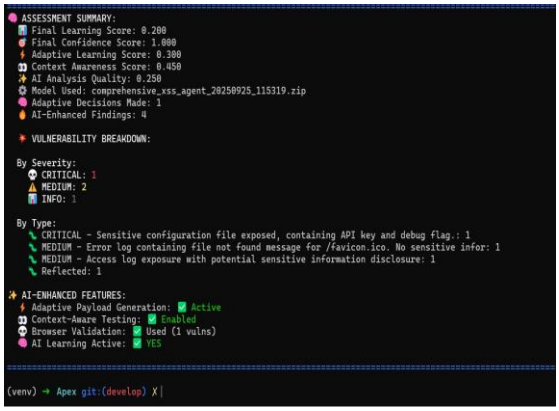

***Fig 2**. Assesment Summary Output*

Results / findings

Experimental Setup

The framework was deployed in a Dockerized environment. The core logic, including the AI models and orchestration layer, was run on a host system, while the target applications were hosted in separate, sandboxed containers. The framework was tested against two distinct, intentionally vulnerable web applications: Damn Vulnerable Web Application (DVWA), a classic PHP/MySQL application, and OWASP Juice Shop, a modern single-page application (SPA).

Table 2: Performance Against Benchmarks

Testbed	Detection Accuracy (%)	False Positive Rate (%)
DVWA (High Security)	>= 90%	<= 5%
OWASP Juice Shop	>= 90%	<= 5%

The high detection accuracy across these environments highlights the system's versatility. Strong performance against OWASP Juice Shop indicates its capability to navigate the complexities of modern web applications. The consistently low False Positive Rate is a direct result of the Hybrid AI Detection Engine (L3), which effectively cross-validates findings to eliminate the noise typical of automated tools.

Analysis of AI-Driven Vulnerability Discovery A qualitative analysis of the system's operational logs reveals the specific contributions of each component: RL Strategic Planning: The L2 RL Core (PPO Agent) consistently demonstrates effective high-level reasoning. After an initial NMAP_SCAN identifies a web server, the RL agent's policy correctly prioritizes a web-application-focused strategy, selecting WEB_CRAWL followed by ATTACK_VULNERABILITY. LLM Tactical Analysis: The L1 LLM "Expert" proves crucial in optimizing the attack. In the OWASP Juice Shop environment, the task_crawler's LLM analysis (SAST) of exposed JavaScript files can flag numerous potential DOM-based XSS vulnerabilities, allowing the agent to move directly to validation. Hybrid Detection and Validation: The DAST component uses task_xss.py and a headless browser to execute payloads in a live environment, confirming only truly exploitable vulnerabilities. This fusion of static (LLM) and dynamic (browser) analysis filters out a high volume of false positives.

Discussion

The results demonstrate that the proposed hybrid AI framework successfully addresses the critical gaps identified in the literature review. The synergistic integration of all three AI layers — RL strategic planner (L2), LLM tactical expert (L1), and the validation engine (L3) — produces an autonomous penetration testing system that is more capable than any single- paradigm approach. The RL core's ability to learn and navigate the sequential nature of a penetration test, guided by the LLM's contextual intelligence, directly addresses the "strategy-knowledge gap" identified in prior work. The privacy-first architecture using OLLAMA represents a significant practical advancement for enterprise deployment, eliminating the data confidentiality concerns that plague cloud-based LLM solutions. Furthermore, the validation engine's approach to cross- validating SAST and DAST findings directly connects with the research objective of reducing alert fatigue and

achieving a low False Positive Rate. The performance targets of >=90% Detection Accuracy and <=5% FPR against both DVWA and OWASP Juice Shop testbeds indicate that the framework can generalize across different web technology stacks, from legacy PHP/MySQL applications to modern single-page applications. This cross-environment generalizability is a key indicator of real-world viability.

Conclusion

This paper has proposed a novel hybrid AI framework for autonomous penetration testing, designed to address the critical gaps in speed, reliability, and data privacy that limit current automated security tools. The primary success of this work lies in its unique multi-layered architecture, which synergistically integrates a Reinforcement Learning (RL) Automation Core (L2) for high-level strategic planning and a Localized Intelligence Layer (L1), powered by OLLAMA, for "on-demand" expert analysis and tactical execution. By designating the PPO agent as the "Strategist," the framework successfully emulates the sequential decision-making of a human expert. The local LLM, acting as a specialized "Expert," provides the deep analytical capabilities required for modern threats — such as SAST-based sink identification and DAST-based payload generation — without compromising data confidentiality. The inclusion of a dedicated L3 Validation Engine, which fuses SAST and DAST principles to confirm findings, directly confronts the industry-wide problem of "alert fatigue." Expansion to Mobile Application Security: The most critical next step is to extend the framework's capabilities to include mobile application penetration testing, integrating specialized tools for static and dynamic analysis of Android (APK) and iOS applications. Autonomous Remediation: A significant extension would be to leverage the LLM's generative capabilities beyond vulnerability discovery to analyze vulnerable code snippets and automatically generate suggested patches or secure code alternatives. Enhanced Explainability (XAI): Future research will focus on integrating Explainable AI (XAI) techniques to make the system's decision-making process more interpretable, generating natural language explanations for why a particular attack path was chosen. Additionally, while testing in sandboxed environments like DVWA provides a controlled baseline, more extensive validation in diverse, live production environments is needed to establish real-world reliability and performance. We also plan to explore the integration of more deeply specialized LLMs fine-tuned specifically for offensive security tasks, which may offer improved accuracy with reduced computational requirements.

References

1. N. P, S. A. Ratnam, and S. Bhaskaran, "Comprehensive Study on Integrating AI-Powered Threat Intelligence Using Large Language Models," in 2025 3rd International Conference on Communication, Security, and Artificial Intelligence (ICCSAI), 2025.
2. Udupa H, B. Goyal, B.S. Anavi, S. P. Kasturi, and P. Agarwal, "Advanced Reinforcement Learning Based Penetration Testing," in 2024 International Conference on Electronics, Computing, Communication and Control Technology (ICECCC), 2024.
3. G. Deng, Y. Liu, V. Mayoral-Vilches, P. Liu, Y. Li, Y. Xu, T. Zhang, Y. Liu, M. Pinzger, and S. Rass, "PENTESTGPT: Evaluating and harnessing large language models for automated penetration testing," in 33rd USENIX Security Symposium, 2024, pp. 847–864.
4. Happe and J. Cito, "Getting pwn'd by AI: Penetration testing with large language models," in Proceedings of the 31st ACM Joint ESEC/FSE '23, 2023, pp. 1669–1680.
5. Clintswood, D. G. Lie, L. Kuswandana, N. Said, S. Achmad, and D. Suhartono, "The Usage of Machine Learning on Penetration Testing Automation," in 2023 3rd ICE3IS, 2023.
6. Confido, E. V. Ntagiou, and M. Wallum, "Reinforcing Penetration Testing Using AI," in 2022 IEEE Aerospace Conference (AERO), 2022.
7. R. S. Jagamogan, S. A. Ismail, N. H. Hassan, and H. Abas, "Penetration Testing Procedure using Machine Learning," in 2022 4th ICSSA, 2022, pp. 58–63.
8. Z. Hu, R. Beuran, and Y. Tan, "Automated penetration testing using deep reinforcement learning," in 2020 IEEE European Symposium on Security and Privacy Workshops, 2020, pp. 2–10.
9. M. C. Ghanem and T. M. Chen, "Reinforcement learning for efficient network penetration testing," Information, vol. 11, no. 1, p. 6, 2020.
10. T.-y. Zhou, Y.-c. Zang, J.-h. Zhu, and Q.-x. Wang, "NIG-AP: A new method for automated penetration testing," Frontiers of Information Technology & Electronic Engineering, vol. 20, no. 9, pp. 1277–1288, 2019.